REMARKS

Applicant has carefully reviewed the Office Action dated August 24, 2006. Applicant has amended Claims 1 and 8 to more clearly point out the present inventive concept and have added new Claims 14 and 15. Reconsideration and favorable action is respectfully requested.

Claims 8-14 stand rejected under 37 C.F.R. 1.75(b) as not being substantially different from Claims 1-7. These claims have been amended and, therefore, they appear to overcome this rejection, the withdrawal of which is respectfully requested.

Claims 1-14 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Hotley* and *Zimmer et al.* This rejection is respectfully traversed with respect to the claims as currently presented.

The claims have been continually rejected over *Hotley*, this being the fifth Office Action in which the Examiner has substantially repeated the same rejection. Applicant believes that the Examiner is incorrectly applying *Hotley* and Applicant will attempt to go through in detail the differences between *Hotley* and the claimed invention and hopefully clear up any issues.

Claim 1 is the primary independent claim. Claim 1 is directed toward a method for protecting a memory space from external access. This memory space is defined as having a plurality of logical portions. These logical portions comprise the portion of the memory space that is to be protected or capable of being protected. In the memory space, and at a predetermined location therein (occupying one of the logical portions), there is stored a plurality of lock bits. The lock bits are defined in a manner that each lock bit is associated with the separate one of the logical portions of the memory space. Figure 13 illustrates this wherein there are provided eight lock bits, one each for a page of memory. The page of memory corresponds to the logical portion and each of the eight lock bits comprises one of the lock bits in the claim. It is noted that the region (1306) in Figure 13 of the current specification constitutes the lock bits that are stored in and comprise a portion of one of the logical portions. As set forth in the specification, these lock bits

just occupy a known location in one of the pages. The lock bits are determinative as to allowing access to the logical portions in response to some external device attempting to perform a memory access operation thereon. The first operation is to detect a request for access to some location in one of the logical portions. The second step is to compare this requested memory access operation with the associated lock bit and then determine if access is allowed thereto for this operation. If allowed, then the particular memory access operation is performed.

Detailed Description of the Operation of *Hotley*

Before addressing the Examiner's rejection, a detailed description of *Hotley* will be undertaken, although these details to some extent have been described before. The *Hotley* reference provides a way to protect access to a memory, particularly a flash card. On the flash card, as seen in Figure 1, the memory card is provided with a plurality of flash memory devices and an access control microprocessor (ACP). This ACP is operable to interface with the memory through a BUS (105) for the purpose of controlling access thereto from an external processor, a host processor. In general, the ACP contains a memory for storing keywords for each block in each memory chip on the memory card. The ACP provides this keyword to the memory chip for the purpose of validation upon creation, upon power up and upon a Write operation when writing to a protected memory location. Each of the memory blocks (103) has independent hardware and logic to control the secure access. The details of the hardware are illustrated in Figure 3.

In Figure 3, there is provided a flash memory block (54) which is described beginning at column 7, line 11 with respect to Figure 3. The memory section is defined as being a flash memory with a memory array 54. Flash memory is well known to be non-volatile memory. This memory section is disposed in conjunction with a security section (103S) that contains additional logic. Illustrated in the memory portion is an output buffer (52), which the Examiner has referred to specifically in the Office Action, which receives an enable signal from an access control memory (43) in the security logic portion (103S) - this being a *volatile* memory. This will be described herein below. However, the output buffer (52) is the buffer that can prevent access to the memory cell at a particular protected block as a result of the contents of access

control memory (43).

The security section (103S) is described beginning at column 8, line 8. The first paragraph in this section is important, as it describes the operation of the access control memory (43). This first paragraph is set forth as follows:

> As seen from FIGURE 3, section 103S includes a security access control unit 30, shown in greater detail in FIGURE 4, a lock write allow storage element 32, *and a volatile access control memory 43* interconnected as shown. The output of the access control memory 43 is applied as an enabling input to output buffer 52 during each memory read cycle when the contents of a byte location of any block of memory array 53 is being read out. That is, a read cycle may occur, however, the data read out is inhibited from passing through output buffer 52 in the absence of the appropriate block's access control memory gating signal. (Emphasis Added)

It can be seen that this access control memory (43) is a volatile memory and it is not a portion of the flash memory. Therefore, it cannot be a portion of any one of the blocks in the array (54). This is a separate and distinct register apart from any addressable location in the memory card. This access control memory (43) contains a plurality of gating signals, one for each block. When the gating signal is set to the logic "0" value, access is allowed to that block when the block is accessed for a Read operation. It is noted that the access control memory (43) is connected to the output of the address latch counter (56). As a result of the access control memory (43) being volatile, the contents thereof are destroyed whenever power is removed and, upon power up, the contents must be returned in order to provide any protection to a particular block via the output buffer (52). Since the uppermost portion of the address constitutes the block address, it is this portion of the address that is input to the access control memory (43). As set forth in the specification, in the paragraph beginning at column 8, line 20, there is provided an address decoder and an output multiplexer. Thus, once the address is decoded, the multiplexer can then select the one of the individually addressable bit storage elements to provide a single output, the enable output, to the output buffer (52). Thus, a Read is performed in response to a Read request, but the output may be blocked by buffer (52) as a result of the contents of access control memory (43).

The security section (103S) is further described in more detail in Figure 4, the operation thereof for allowing access also described in more detail. The memory array is divided into two sections, a lock bit section (54a) and a data section (54b) (column 8, lines 47-50.) Each block of memory contains a plurality of rows, in this embodiment, 8K rows. Each row contains 8 byte locations, B0-B7. In this disclosure, an additional bit location has been added to each row to form the lock bit section (54a). Thus, each row has an available location in which to store a single lock bit.

The specification of *Hotley* further goes on to illustrate how these sections are addressable via an address latch counter (56). This defines the first register section that is operable to store a block select portion of the address, this being the upper MSBs. The second section, a row select section, selects one of the rows therein. A third section, the BYTE SEL section is defined as the third register section and this is used to store the least significant group of address bits designating which byte of an addressed row is being addressed. It can be seen that there is a data register/output multiplexer (53a) that is interfaced with each of the bytes B0-B7. The least significant bits are illustrated as selecting which byte is to be output. However, it is noted that there is no indication as to how the additional single bit is actually selected through the address. It is expected that addressing of the row with the middle significant bits will access all bits therein and allow at least output of the data in this bit location. However, there is no detail as to how this bit location is written to or read from.

The lower portion of Figure 4 is directed toward the operation of the access control during a validation process to either initially load the lock bits for determining and setting the contents of access control memory (43). This access control, as set forth in the specification and which will be described in more detail herein below, is a control method wherein a keyword can be received from the ACP (10). The length of this keyword can be equal to the number of available lock bits in a particular column in a particular block, i.e., the section (54a). As noted herein above, there is provided one lock bit for each row and, therefore, there can be as many lock bits in the keyword as there are rows in a block. However, it is not necessary to provide a keyword of this length. It can be equal to or less than the number of the available rows in a

block. The only important thing is the arrangement of the bits in the keyword. The first bit stored is a logic "0" or a logic "1" depending upon whether a particular block is to be protected. If it is a logic "0," this indicates that the block is to be protected and if it is a logic "1", this indicates that the block is not to be protected. Note that a determination is made by the ACP (10) as to what this value should be. In general, as will be described herein, when protecting a block, it is necessary to write a logic "0" in the first lock bit location LMB0. Thereafter, the key bits are sequentially stored in the lock bits LMB1, LMB2. . . LMBn. The last 7 bits are logic "1" bits such that the end of a particular keyword can be detected. This means that the remaining keyword bits can have no more than six consecutive logic "1" values.

There are a plurality of commands that are generated, a Start command, a Step command, and an End command. The Start command basically starts an operation (the only operation that requires any type of compare), the Step command steps through the various lock bits and the End command basically ends the process. As will be noted herein, there are three different processes, one for loading the lock bits and setting the value of the enable bits in that access control memory (43), one for loading the volatile access control memory (43) on power up and one for accessing the memory for a Write operation. Access to the memory during a Read operation does not involve these lock bits in the array (54a) or the keywords.

In general, during the Step operation in the power up operation, the key bits are input one at a time and the bit compare logic block (30-1) utilized to compare the two bits. The results of that comparison are input to an accumulator (30-2) to accumulate the results, i.e., were all of the steps a hit? Also, a counter is provided for counting the number of successive logic "1" bits occurring in a string of lock bits and is used to detect the end of stored key value. The access control memory (43) is operable to receive the lock bit directly, for operations wherein the lock bit is stored directly therein, specifically when the LMB0 bit is accessed, for the initial state of the access control memory. As set forth in column 10, beginning at line 10, the End instruction is performed once the ACP (10) determines that the key validation operation is terminated, and then it tests the output of the accumulator and the counter. When both are correct, it sets the enable bit in the particular address location in the access control memory (43). Again, this is a

volatile memory and must be set in order to provide the correct enable signal on the output thereof for Read operations. Again, and Applicant will note this a number of times herein, this access control memory (43) is not a part of the non-volatile flash memory array nor are the enable bits stored in the non-volatile memory array in a section (54a) utilized to generate the enable signal.

The description of the actual operation begins on column 10, line 34. As noted herein above, there are three operations, one with respect to Figure 6a, one with respect to Figure 6b and one with respect to Figure 6c. In the initial operation, that associated with Figure 6a, there is illustrated a process for initially loading the lock bits into the memory. The entire memory is first erased and, thereafter, each block must be written. There is provided no protection against this erasure. There is a configuration stored in the ACP that determines whether a block is to be protected or not to be protected. (Column 11, lines 53-57). The operation set forth in the flow chart of Figure 6a is that the first block is addressed and then erased. The ACP then determines from the configuration information if this block is to be protected. If not, then the program will flow to the block (612) which will execute the End command and sets the associated bit in the access control memory (43). If it was not a protected block, this would be set to a logic "1." In general, when a flash memory is erased, all of the bits are at a logic "1" and, as such, nothing needs to be written at this point. Thereafter, there is a pre load operation in block (614) wherein the various block contents are loaded thereto. This, of course, depends upon the operation and the applications that need to be pre loaded to the memory.

If, however, at block (604) it were determined that the block were to be protected from the configuration information in ACP (10), the first lock bit position LMB0 has a logic "0" written thereto and then the program proceeds to store the bits of the keyword from ACP (10) in the lock bit positions in the section (54a) for the given block. This continues until the ACP determines that seven consecutive bits have been encountered in the key sequence and then it initiates the End command at block (612). This, again, is where the particular bit in the access control memory (43) is set, i.e., it is loaded into the volatile memory, i.e., protection must be validated prior to protection being enabled. This continues until all blocks are evaluated. This

operation in Figure 6a is one that is typically carried out during fabrication by the manufacturer. There is no compare operation with the contents of the memory required here, as the block is erased prior to beginning.

The next section is the operation during power up is described in figure 6b. This is necessary, as noted herein above, because the access control memory (43) has undetermined data therein, since it is a volatile memory. The first thing to do is to clear the access control memory (43) and perform various resets. The first block of memory is then addressed and the Start instruction initiated. It is important to remember that, at this time, the lock bits are stored in the memory location array (54a). The program then proceeds to block (626) and determines if the block is protected, i.e., is LMB0 is a logic "0". (Column 13, lines 1-4.) This basically protects the memory if it is to be protected, i.e., if LMB0 is a logic "0" this means that it is protected, and if it is a logic "1" this means that it is unprotected. If it is protected, then the next step would be to execute the Step instruction to test all of the lock bits and then compare them until the entire keyword was processed. Once processed, then the End instruction is executed and the associated enable bit location in the access control memory (43) set to a logic "0". (Column 13, lines 61-column 14, line 5.) Thus, it can be seen that the power up operation merely validates each block by comparing its associated lock bits with the keyword to determine if it is protected or unprotected. If protected, then it must set the associated enable bit location in the access control memory (43) to the appropriate logic level. This is the only process that involves any compare operation of the stored lock bits.

Referring now to Figure 6c, there is illustrated the operation after power up for a Write operation. This is described beginning at column 14, line 11. This particular operation is an operation that is performed whenever a Write operation is desired. A Write request does not have to be allowed, as it merely erases an entire block for the purpose of writing. Thus, even if a block is protected, it will be erased, as is the case with an unprotected block. When erased, as set forth in the specification, and as is well known in the art, all of the bits are written to a logic "1." As also described in the specification, if the lock bit LMB0 remains at a logic "1" state, this indicates that it is a block that is not to be protected. Since the contents of the access control

memory (43) will allow a Read operation, but are not utilized for any kind of data Write operation, this particular access control memory block (43) is not involved in a Write operation. However, some accommodation must be made when writing to a protected block, as the Write operation is a destructive process and actually erases the lock bits associated with that protected block. Thus, that is the purpose of the process of Figure 6c - to restore the lock bits. It can be seen that the first thing that occurs is that the selected block is addressed, as indicated by block (640), and then this block is erased, thus erasing the lock bits. This is the purpose of modifying the operation of the Start and Step instruction is to cause the bits presented by such instructions to be written into the lock bit positions in lieu of being compared to them. (Column 11, lines 22-34.) The next step in the process is decision block (644). This is described at column 14, beginning at line 24, the disclosure of which is set forth as follows:

> Next, as indicated by block 644 of FIGURE 6c, ACP 10 determines from the stored configuration information if the erased block is to be protected. If it is protected, the ACP 10 will execute a start instruction. Since the write lock allow flip-flop 32 is set, it will modify the operation of the start instruction so that it causes a binary ZERO to be written into the first bit position (LMB0) of the lock memory area of the selected block in lieu of performing a compare operation. Next, as indicated by blocks 648 and 650, the ACP 10 will execute a number of step instructions for writing the bits of the key value presented by the step instructions into the lock bit positions of the selected block until all of the bits have been written, signaled by the detection of 7 consecutive ONE bits.

It can be seen that this decision block (644) proceeds along the "Yes" path only if the block is to be protected, i.e., the lock bits must be written back into the memory. However, if the block is not to be protected, i.e., it was originally designated in the ACP (10) as not being protected, then there is no reason to rewrite the lock bits, as they are all a logic "1". This description is set forth as follows beginning at column 14, line 45:

> If the block is not to be protected as per the configuration information, the execution of any instruction will inhibit the writing of lock bits by causing the resetting of the lock write allow flip-flop 32. Next, as indicated by block 654 of FIGURE 6c, the ACP 10 executes a start instruction which operates in the normal

> way to transfer the 0th lock bit read out from the selected block and strobe it into the ACM storage element associated with that block.

It can be seen from this portion of the disclosure that the configuration information outside of the memory and part of the ACP (10) is utilized to determine whether the block is to be protected or not to be protected prior t. It states that the execution of any instructions will "inhibit" the writing of lock bits as it causes a reset of flip-flop (32). Essentially, the system goes around the lock bit rewrite operation and merely loads the block contents at block (656), although the actual writing of information is not discussed. The specification does set forth that the LMB0 lock bit is read out from the selected block and strobed to the access control memory (43) when a block is unprotected, but this does not seem to be a necessary step, as the access control memory (43) was never cleared and the storage element should already be at a logic "1." However, what is important to note is that, when a Write operation is to be performed, an erase of the memory is provided which sets all of the lock bits at a logic "1" state in that associated block, there is no clearing of the access control memory (43) and, as such, there is no requirement to check that particular bit location in access control memory (43) in order to determine whether a Write operation is allowed on that particular block. The Write operation is initiated regardless of the protection level and then the ACP (10) examined only to determine if lock bits need to be rewritten. There is no provision therein that access for a Write will be granted or will not be granted but, rather, a process for writing the contents of the data with or without writing the lock bits back into the portion of the memory. Of course, nothing can be written to the memory unless the ACP runs the program associated therewith.

As a summary of the operation, Applicant notes that the purpose of this particular memory is to provide a highly producible and programmable key evaluation system for a memory (Column 14, lines 56-57.) The operation is to provide a "volatile" memory for containing a plurality of access control bits which are then operable to control a Read operation. (Column 8, lines 12-19.) Thus, there is provided the ability to protect certain blocks of memory from being read during normal operation. Whenever a Write operation is to be performed, in accordance with Figure 6c, what happens is that the entire block is erased regardless of the

protection level and then a procedure is undertaken to go to a separate portion of the memory, not the memory array itself or a logical portion of the memory but, rather, the ACP (10), in order to rewrite the lock bits if this particular erased portion was protected. Therefore, the Read operation is controlled by the access control memory (43), a volatile memory that is not a portion of the memory array, and the Write portion is controlled by the ACP (10), this having its own storage element (10-2), separate and apart from the memory array associated with the protected blocks.

History of the rejections

The Examiner has, including the present rejection, rejected the claims five times with the *Hotley* reference as the primary reference. Applicant, for clarification purposes, will go through each of the rejections to show that the claims at the time of the rejection were not anticipated or obviated by *Hotley* as the Examiner has set forth.

In the rejection of Claims 1-7 in the Office Action of December 24, 2003, the First Office Action, Claim 1 required the step of "storing in a location in memory a plurality of lock bits, each associated with a separate logical portion of the memory space and determinative as to the access thereof for a predetermined operation thereon . . ." Further, the claim required there to be the step of "comparing the requested operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested operation." The Examiner basically stated the following in the Office Action at paragraph 6:

> With respect to claim 1, Hotley discloses:
>
> storing in a location in memory a plurality of lock bits, each associated with a separate logical portion of the memory space and determinative as to the access thereof for a predetermined memory access operation thereon, as shown by the lock bits (item (54a)) for each row (54b) of the memory array in figure 4;
>
> detecting a request for access to a desired location in the memory space for operating thereon, as shown in figure 6b;
>
> comparing the requested memory access operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested memory access operation, and performing the requested memory

access operation if allowed, as shown by protection determination and ensuing execution starting in step 626 in figure 6b.

It can be seen that the claim requires that the requested operation be compared with the associated lock bit to determine if access is allowed. The Examiner states that *Hotley* discloses such but, it is important to note that there is no comparison of the lock bits at any time for the purpose of allowing or disallowing access. The claim specifically requires that the comparing operation be compared with the "associated lock bit in the *associated logical portion* in order to determine if access is allowed". There is no Read operation during access of the memory that occurs in any manner within *Hotley* utilizing the lock bits in the array (54a). All that is required is that the access control memory (43) be accessed for a Read operation. This is not a portion of the memory space. This access control memory (43) is not in the memory space. The Examiner specifically refers to the array (54a) as the area wherein the lock bits are stored. Clearly, there are lock bits stored in a portion of each block and this does constitute a portion of the memory array. However, there is no comparison made with any bit or sequence of bits stored in the memory array for the purpose of allowing access or for determining if access is allowed for a requested operation. These bits are for validation purposes only. The output buffer (52) will allow a Read operation to occur if the enable bit in access control memory (43) is set, this being a volatile memory that is not a portion of the memory array. For a Write operation, no protection is provided. Thus, Applicant believes, as noted in the response, that *Hotley* does not disclose the portions that the Examiner had suggested.

Applicant notes that the Examiner has referred to step 626 in figure 6b. As noted herein above, figure 6b is the validation process of a power up. There is no request to write or read from the memory but, rather, all the process of figure 6b does is determine if it is protected or not protected and, if protected, set the access control memory (43) bit to the correct bit. There is no attempt to access any particular block for a memory access operation thereon. As such, Applicant believes that the Examiner is referring to the wrong operation in this rejection with respect to the comparison step.

In the response filed on May 24, 2006, Applicant amended the claims to clarify that the requested operation was a memory access operation.

In the second Office Action, dated June 15, 2004, the Examiner again rejected Claims 1-7, utilizing *Hotley* as the primary reference. The Examiner's statements with respect to *Hotley* are set forth in paragraph 5 as follows:

> With respect to claim 1, Hotley discloses:
>
> storing in a location in memory a plurality of lock bits, each associated with a separate logical portion of the memory space and determinative as to the access thereof for a predetermined memory access operation thereon, as shown by the lock bits (item (54a) for each row (54b) of the memory array in figure 4;
>
> detecting a request for access to a desired location in the memory space for operating thereon, as shown in figure 6b;
>
> comparing the requested memory access operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested memory access operation, and performing the requested memory access operation if allowed, as shown by protection determination and ensuing execution starting in step 626 in figure 6b.

This is substantially, if not exactly, identical to the first rejection of June 15, 2004. This rejection refers to the claim amendments filed by Applicant on May 24, 2004. The amendments thereto did not change the aspect wherein the plurality of lock bits were stored in the memory and that the compare operation compared an associated lock bit "in the associated logical portion" for determining if access is allowed.

The Examiner indicated that the validation procedure of *Hotley* reads upon the claims, but the Examiner did not address the fact that the lock bits associated with the validation process are not used for memory access operations.

The Applicant amended the claims to change some minor aspects thereof such as the requested operation now being a requested "memory access" operation. However, the remainder of the claims still remains the same.

The Examiner again rejected the claims in a third Office Action dated December 23, 2004. The Examiner stated in paragraph 6 that:

> With respect to claim 1, Hotley discloses:
>
> storing in a location in memory a plurality of lock bits, each associated with a separate logical portion of the memory space and determinative as to the access thereof for a predetermined memory access operation thereon, as shown by the lock bits (item (54a) for each row (54b) of the memory array in figure 4;
>
> detecting a request for access to a desired location in the memory space for operating thereon, as shown in figure 6b;
>
> comparing the requested memory access operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested memory access operation, and performing the requested memory access operation if allowed, as shown by protection determination and ensuing execution starting in step 626 in figure 6b.

In this Office Action, the Examiner did not further elaborate on *Hotley*. Applicant amended the claims to require at least two different memory access operations.

The Examiner again rejected the claims, on May 12, 2005. In paragraph 11 thereof, the Examiner stated the following:

> With respect to claims 1 and 8, Hotley discloses:
>
> storing in a location in memory on one of the logical portions thereof a plurality of lock bits, each of the lock bits associated with a separate one of the logical portions of the memory space and determinative as to the access thereof for a predetermined memory access operation thereon, as shown by the lock bits (item 54a) located in a separate logical column, for each row (54b) of the memory array in figure 4;
>
> detecting a request for access to a desired location in the memory space for operating thereon, as shown in figure 6b;
>
> comparing the requested memory access operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested memory access operation, and performing the requested memory access operation if allowed, as shown by protection determination and ensuing executing starting in step 626 in figure 6b;

there being at least two different memory access operations, as taught in column 9, lines 2-6.

The only addition that the Examiner added to this portion of his rejection was the aspect that there are at least two different memory access operations, referring to column 9, lines 2-6. That portion of the specification in *Hotley* is set out as follows:

> A multiplexer/demultiplexer circuit 53a which includes the circuit of block 53 is used to select the byte location to be written into or read as a function of the least significant address bits stored in address latch counter 65.

This portion refers to nothing more than the input/output multiplexer (53a) for selecting the byte locations for reading data therein or extracting data therefrom. However, the claim is directed toward determining from the state of the lock bit whether such access is allowed. This portion of the specification has nothing to do with that. As noted herein above in Applicant's detailed analysis of *Hotley*, the Read operation does not require any analysis of lock bits in that particular block but, rather, it looks to the contents of the access control memory (43). For a Write operation, erasure is always allowed, but the configuration information ACP (10) is what is accessed prior to writing to the memory. Therefore, this particular multiplexer (53a) has nothing to do with the protection operation or for allowing access or for not allowing access.

The claims that were under consideration for this Office Action were the claims provided in the previous response to the December 23, 2004 rejection and they merely set forth that the memory access operation was a "predetermined type of" memory access operation. The remaining portion of the claims with respect to the lock bits stored in the memory and that the comparison operation required comparing the associated lock bit in the associated logical portion with the requested memory access operation remain substantially the same as the originally filed claims.

Current Office Action

In response to the Office Action of May 12, 2005, Applicant filed a Response on June 11, 2006 further clarifying some aspects of Claim 1 with respect to the memory comprising a memory space which in the logical portions were included. Again, Applicant made these amendments in an attempt to clarify the claims and explain the distinctions between *Hotley* and the claimed invention. This resulted in the current Office Action in which *Hotley* is again utilized as the primary reference. In paragraph 10, the Examiner again described *Hotley* as disclosing substantially the invention noting that *Hotley* failed to specifically disclose using the lock bits to determine if the requested predetermined operation of access is allowed. This portion of the rejection with respect to Claim 1 and Claim 8 is set forth as follows:

> With respect to claims 1 and 8, Hotley discloses:
>
> storing in a location in memory on one of the logical portions thereof a plurality of lock bits, each of the lock bits associated with a separate one of the logical portions of the memory space and determinative as to the access thereof for a predetermined memory access operation thereon, as shown by the lock bits (item 54a) located in a separate logical column, for each row (54b) of the memory array in figure 4;
>
> detecting a request for access to a desired location in the memory space for operating thereon, as shown in figure 6b;
>
> comparing the requested memory access operation with the associated lock bit in the associated logical portion and determining if access is allowed for the requested memory access operation, and performing the requested memory access operation if allowed, as shown by protection determination and ensuing executing starting in step 626 in figure 6b;
>
> there being at least two different memory access operations, as taught in column 9, lines 20-6.

Applicant notes that this rejection again is substantially the same rejection as the rejection of May 12, 2005.

The Examiner, in paragraph 11, has made arguments with respect to Applicant's previous responses. These remarks by the Examiner are set forth as follows:

> As per remark, Applicant's counsel asserts that (a) "*Hotley* stores the memory locations in an area of the memory that cannot be protected, i.e., the lock

byte protects itself." (amendment, page 6, fourth paragraph).

> First of all, it should be noted that *Hotley* discloses the stored memory locations can be protected wherein *Hotley's* figure 4 again shows the lock bits in the separate logical column from the rest of the data bits. In addition; in considering a 35 U.S.C. § 103 rejection, it is not strictly necessary that a reference or references explicitly suggest the claimed invention (this is tantamount to a 35 U.S.C. § 102 reference if the modifications would have been obvious to those of ordinary skill in the art. It has been held that the test of obviousness is not whether the features of a secondary reference may be bodily incorporated into the primary references' structure, nor whether the claimed invention is expressly suggested in any one or all of the references; rather, the test is what the combined teachings of the reference would have suggested to those of ordinary sill in the art. See In re Keller et al., 208 U.S.P.Q. 871. In addition, Examiner further recognizes that references cannot be arbitrarily combined and that there must be some reason why one skilled in the art would be motivated to make the proposed combination of primary and secondary references. In re Nomiya, 184 USPQ 607 (CCPA 1975). However, there is no requirement that a motivation to make the modification be expressly articulated. The test for combining references is what the combination of disclosures taken as a whole would suggest to one of ordinary skill in the art. In re McLaughlin, 170 USPQ 209 (CCPA 1971). *Zimmer et al., Sharma et al, Wolrich et al.* and *Hotley* references are evaluated by what they suggest to one versed in the art, rather than by their specific disclosures. In re Bozek, 163 USPQ 545 (CCPA) 1969. In this case, the *Zimmer et al., Sharma et al., Wolrich et al.* references were used to provide evidence of separate lock bits are utilized for both read and write access type which is known to be required in the system of *Hotley* in order to arrive at Applicant's current invention. The 35 U.S.C. § 103 rejection based on said combination is therefore deemed to be proper.

The Examiner specifically states that the "stored memory locations can be protected wherein *Hotley's* figure 4 again shows the lock bits in a separate logical column from the rest of the database." As noted herein above, these lock bits are not utilized to control access; rather, the only portion that controls access for a Read operation involves the enable bits that are stored in the access control memory (43). Figure 4 illustrates the memory section and there is only a reference on the bottom right wherein there is an output to the access control memory (43). However, this is only utilized to update the contents and not to control access. The portion of the security block in the lower right hand corner, i.e., the security access control unit (30), is not utilized for access to the data when unprotected. This portion of the memory is only utilized to

initially determine which sections are protected and set the bit therefore to a logic "0" in the access control memory (43). Thus, the Examiners statement that the stored memory locations can be protected with reference to figure 4 and the lock bits shown therein, is not relevant to the claim, as the claim is directed toward lock bits that allow access and these can only be the ones stored in the access control memory (43). During normal operation for controlling access to the memory, there is no access of the lock bits in the memory in section (54a). Thus, Applicant believes that this is an incorrect statement on the Examiner's part and makes and constitutes incorrect analysis of the operation of *Hotley* with respect to the claims.

Obviousness Analysis

The Examiner has analyzed, with reference to *In re Keller et al. 208 U.S.P.Q. 871,* the test for obviousness. However, this is a relatively old case, a 1981 case. Although the general premises are correct, it is preferable that the more recent cases be examined. For example, the most recent case by the Federal Circuit on obviousness is *In re Kahn, 78 U.S.P.Q. F.2d, 1329, (Fed. Cir. 2006).*

In general, MPEP §2142 specifies that:

> The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness. If the examiner does not produce a *prima facie* case, the applicant is under no obligation to submit evidence of nonobviousness.

In regard to what an examiner must show in order to establish a prima facie case of obviousness, MPEP §2142 further explains that:

> To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. . . . Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.

In regard to what an examiner must do in order to meet the first criterion for a prima facie rejection, MPEP §2143.01 specifies that:

> Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art.

Thus, it is required that the Examiner make a *prima facie* case that either the single prior art reference or the combined prior art references teach or suggest all of the claimed limitations. The court in *Kahn* set forth an analysis with respect to whether subject matter would have been non-obvious under § 103. This is set forth in *Kahn* at 985-986 citing *In re Rouffet, 149 F.3d 1350 (Fed. Cir. 1998)* as follows:

> In assessing whether subject matter would have been non-obvious under *§ 103*, the Board follows the guidance of the Supreme Court in Graham v. John Deere Co. The Board determines "'the scope and content of the prior art,'" ascertains "'the differences between the prior art and the claims at'" issue, and resolves "'the level of ordinary skill in the pertinent'" art. *Dann v. Johnston, 425 U.S. 219, 226, 96 S. Ct. 1393, 47 L. Ed. 2d 692 (1976)* [**17] (quoting *Graham, 383 U.S. at 17*). Against this background, the Board determines whether the subject matter would have been obvious to a person of ordinary skill in the art at the time of the asserted invention. *Graham, 383 U.S. at 17*. In making this determination, the Board can assess evidence related to secondary indicia of non-obviousness like "commercial success, long felt but unresolved needs, failure of others, etc." *Id., 383 U.S. at 17-18*; accord *Rouffet, 149 F.3d at 1355*. We have explained that
>
> > to reject claims in an application under *section 103*, an examiner must show an unrebutted prima facie case of obviousness . . . . On appeal to the Board, an applicant can overcome a rejection by [*986] showing insufficient evidence of prima facie obviousness or by rebutting the prima facie case with evidence of secondary indicia of nonobviousness.
>
> *Rouffet, 149 F.3d at 1355.*

As such, the Examiner's statement that the test for combining references is what the combination of the disclosures taken as a whole would suggest to one of ordinary skill in the art must require some type of analysis as to how a particular reference is analyzed. The Examiner has merely stated that the additional references to *Zimmer*, *Sharma* and *Wolrich* provide evidence of separate lock bits for Read and Write operations "which is known to be required in order to arrive at Applicant's current invention." This statement ignores the deficiencies in *Hotley* relative to the fact that the stored lock bits are not used for permitting access to the memory.

To further analyze this, one must look at the *Hotley* reference and determine what the *Hotley* reference suggests. All the *Hotley* reference suggests is that lock bits can be stored in memory for the purpose of validating memory upon initially loading the memory with program instructions upon creation of the memory card, upon power up for the purpose of writing lock bits back to protected memory if necessary during a Write operation. However, during operation, there are no lock bits in the memory locations that are examined as set forth by the Examiner. Again, the Examiner only refers to the lock bits in the memory array (54a) for the purpose of supporting this rejection.

The *Zimmer* reference is utilized to cure the only deficiency in the reference noted by the Examiner, that being that *Hotley* "fails to specifically disclose using the lock bits to determine if the requested predetermined type of access is allowed." Applicant contends, however, that *Hotley* does not suggest utilizing the lock bits stored in the memory for the purpose of protecting access to the memory; rather, as described in detail herein above, the only enable bits that are utilized for that purpose in a Read operation are the enable bits stored in the access control memory (43), a volatile memory that can not be part of the memory array.

The *Zimmer* reference has been provided with reference to figure 2b at column 4, lines 40+, for the purpose of indicating that both a Read lock bit and Write lock bit are provided that would lock access to the memory block for either a Read access type or a Write access type of operation. However, *Zimmer* at column 4, lines 4-14 states that the firmware hub includes an integrated command user interface to request access to locking, programming and erase

operations. This specifically utilizes a locking register (250) shown in figure 2b. It also states in column 4, lines 42-43 that the "block locking register (250) is part of an Intel ® FWH. . ." which indicates that the locking register is not necessarily part of the flash memory. In figure 2b, it specifically refers to the locking register as being "for" the flash memory. Further, it notes that the flash memory is part of the Intel RTM (rear transmission module) and that this register is part of this RTM firmware hub. There is some inconsistency in that the specification sets forth that the detailed flow diagram of figure 4 illustrates an operation for programming the locking register "in flash memory 107" to lock the block in the flash memory. However, it is not clear where the locking register is disposed. Even if it is disposed in the memory, this is a "virtual" lock system wherein the lock bits can be changed upon a reboot. With respect to the description of the flow chart of figure 4, it is stated that "for instance, if the lock bit (254) is set and the Write lock bit (2560 is already set, the bits cannot be changed *until computing system 100 is restarted*." (Emphasis added.) (Column 5, lines 39-41.) Thus, there is no disclosure in *Zimmer* that the lock register (250) is disposed within one of the logical portions associated with any of the lock bits. All that is stated is that there is provided a Write lock bit (256) that, if set in the locking register (250), results in disabling of a Write to the associated block in flash memory. As such, *Zimmer* does not cure the deficiencies noted herein above with respect to *Hotley* in that *Hotley* does not describe a register or a lock bit that is stored within the logical portions of the memory to be protected. The reason for this is that figure 2b in *Zimmer* illustrates that there is a Write lock section (256) that prevents Writes to a particular logical portion of the memory. However, these Write lock bits can be changed, i.e., data can be written to this area of the memory, even though the virtual lock still exists on the memory. Thus, this would teach away from including any kind of locking register as a portion of the logical portions of the memory to be protected. Thus, Applicant believes that the combination of *Zimmer* and *Hotley* do not disclose the incorporation of lock bits into the logical portion of the memory that is to be protected. Further, *Zimmer* actually teaches away from doing such, as it is necessary for the computing system to be able to change the Read lock and Write lock bits if the lock down bit is not set. Even when the lock down bit is set, upon a reboot process, this can be changed and there is no indication that the memory must be erased in order to do such.

The Examiner has also utilized the *Sharma et al.* reference in combination with *Hotley* and to show the use of a Read lock bit which would lock access to the memory block from a predetermined access type, wherein the access type lock is a Read. The *Sharma* reference is a reference that is concerned with a cache memory system and provides therein a status line to lock access to the memory during a request if an outstanding request exists. This is a system that provides for multiple controller units (102), each having a separate memory bank and a memory controller. The cache unit is separate and apart from the memory bank in each of these systems. There is the possibility that multiple requests can be made to the memory from different locations. Therefore, a direct memory access (DMA) is utilized to control access to the memory through the various cache units. In order to do this, there must be some type of "lock" placed on a request to ensure that an original requestor receives data in the cache, which data in the cache must reflect the actual data in the memory. If one requestor is using this cache data, another requestor can not obtain the data or operate on that data. The reason for this is that the data is in the possession of one requestor such that a second requestor must wait. As such, this particular block to a Read access of a memory is in a separate system apart from the memory itself therefore, this particular Read status cannot be stored in a location in the memory space on one of the logical portions nor are there provided lock bits, one associated with a particular one of the logical portions. The particular Read status byte is a temporary Read status byte that is only associated with an address location in the memory for a temporary time. This is not stored in the memory and does not cure the deficiencies noted herein above with respect to *Hotley*, as *Hotley* discloses the enable bits in the access control memory (43) apart from the actual flash memory itself. Therefore, the combination of *Hotley* and *Sharma* are not believed to anticipate or obviate the invention as set forth in Claim 1.

The Examiner has also combined the *Wolrich* reference with *Hotley* for the purpose of disclosing the use of a Read lock bit which would lock access to the memory lock from a predetermined access type, wherein the access type lock is a Read. The Examiner specifically refers to figure 3 of *Wolrich et al.*

In *Wolrich*, as part of a controller (26b), there is provided a content addressable memory

(CAM) to provide look-ups of Read locks. In general, the operation of this locking mechanism is illustrated with respect to figure 3, the discussion of which begins at column 5, line 1. Whenever a Read request is generated for a portion of the memory, this Read is allowed. When there is a Read request for a particular location in memory, then the command decoder and address generator enters a lock into the lock look-up device (142), the CAM. This is such that subsequent Read lock requests will find the memory location locked. In order to read a location, the system must first look to see if that memory location is locked for some reason. The only way to unlock this is to send an unlock request for that memory location. Thus, the controller operates, when accessing a memory, to access the lock look-up device (142) to determine whether a memory location is already locked. If so, it then waits for the memory location to be unlocked. Thus, there is no particular lock associated with and stored in any portion of the memory but, rather, there is merely some type of association in a memory separate and remote from the location to potentially be locked that keeps track of existing Reads, such that some system can gain ownership of a particular location in memory and prevent others from accessing that memory. This allows multiple requests form different systems to a given memory to be handled. Therefore, as was the case with the *Zimmer* and *Sharma* references, there is provided no logical portion of the memory that contains lock bits, each associated with one of the logical portions of memory that can contain lock bits that will prevent access thereto. The system provides nothing more than a separate and distinct CAM for temporarily storing an address of a particular location. *Sharma* and *Wolrich* are very similar in this aspect. Applicant believes that the *Wolrich* reference does not cure the deficiencies herein above with respect to *Hotley*, as *Hotley* does not disclose nor suggest that any lock bits that prevent a memory access operation be stored in the logical portion or portions of the memory which are to be protected.

The Examiner has stated in paragraph 28 (misnumbered and out of sequence Office Action), the Examiner had indicated that Applicant had alleged that the lock bits in *Zimmer*, *Sharma* and *Wolrich* are not contained in a separate logical portion of the memory space, but are contained in a separate memory device altogether. This certainly is the case. The Examiner has indicated that the claims do not limit the memory space to a single physical medium. The Examiner has stated that "only a separate logical portion of the memory space is claimed. As

such, a separate physical memory medium logically associated with the data array would read upon the claim language." However, the claims state that the lock bits must be stored in a location in the memory "on one of the logical portions thereof" and that the lock bits actually are operable to lock the logical portion in which the lock bits are shared. Therefore, it is clear from the claims that the lock bits must be able to protect the logical portions in which the lock bits are stored. None of the references to *Hotley*, *Zimmer*, *Sharma* or *Wolrich* disclose such limitation in the claim, nor do they suggest such limitation in the claim. The *Hotley* reference specifically does not utilize the lock bits in the memory for the purpose of restricting access for any operation, as there is no comparison required for utilizing the lock bits in the memory array for either Read access, Write access or memory access operation.

The Examiner had noted with respect to the application of *Hotley* to Claims 5 and 12 that *Hotley* disclosed that, in the step of comparing, the lock bits are first read from the known location in the memory and then this Read location is *utilized* to read the lock bits from memory. The Examiner specifically refers to column 13, lines 24-35. However, this portion of the specification is referring to the step instructions for incrementing through the stored lock bits. This portion of the specification is that with respect to the power up process that requires the non-volatile access control memory (43) to be updated. This does not relate to any access to the memory, as it actually allows access to the memory. All that is occurring in this portion of the specification is a validation sequence to ensure that the key bits stored in the ACP (10) are identical to the bits stored in association with a particular block and, if so, then the access control bit in the access control processor (10) is set. Again, as described herein above, this portion of the specification does not relate to allowing access to a particular memory location upon a memory access operation.

The Examiner also sets forth with respect to claims 6 and 13 that *Hotley* discloses a predetermined operation being an erase of the lock bits. However, there is no requirement that any examination of the lock bits be made prior to an erase operation. An erase operation is always allowed. The only aspect is that, after erased, any writing back to the particular location must also involve writing of the lock bits thereto. Therefore, the erase would be allowed and

then the Write would be allowed. No lock bit is examined to provide such erasure.

With respect to the Examiner's comments with respect to claims 7 and 14, in view of *Hotley*, the Examiner refers to the teaching in column 14, lines 15-20. This portion of the specification is directed toward the power up process again, and this is nothing more than the Read/Write operation wherein a sequential address is placed into the address register, that block addressed, and an erase (block erase) performed. This utilizes the most significant address bits to erase an entire block. However, the claim specifically refers to the fact that the locations erased not have the lock bits contained therein. However, it is very clear that each block has lock bits associated therewith and erasure of that particular block results in erasure of the lock bits. Therefore, this portion of the specification with respect to erase is not relevant to this particular claim. Further, the erase operation is not an operation that is allowed or denied by the lock bit, as required by Claim 6.

The amendments made to Claim 1 do not change the limitations that were originally in the filed Claim 1. These limitations are not believed to have ever been anticipated by *Hotley* and Applicant believes that *Hotley* does not teach or suggest the limitation of including the lock bits, stored in the access control memory (43), into the memory in a logical portion that is to be protected or associated with a lock bit. As such, Applicant respectfully requests withdrawal of the 35 U.S.C. § 103 rejection with respect to *Hotley* in combination with any of the references. Applicant believes that the Examiner has not established a *prima facie* case of obviousness either using *Hotley* alone or in combination with the other references.

With respect to claims 8-14, Claim 8 has been amended to substantially incorporate the limitations of Claim 1 with the exception that there are not required two memory access operations to be protected. Applicant believes that *Hotley* does not describe the inclusion of the lock bits in logical portions of the memory that can be associated with the lock bits and, thus, can be protected. Therefore, this limitation is not required in order to distinguish over the combination of *Hotley* with any of the other cited references.

Applicant has now made an earnest attempt in order to place this case in condition for

allowance. For the reasons stated above, Applicant respectfully requests full allowance of the claims as amended. Please charge any additional fees or deficiencies in fees or credit any overpayment to Deposit Account No. 20-0780/CYGL-24,962 of HOWISON & ARNOTT, L.L.P.

Respectfully submitted,
HOWISON & ARNOTT, L.L.P.
Attorneys for Applicant(s)

/Gregory M. Howison, Reg. No. 30,646/
Gregory M. Howison
Registration No. 30646

GMH/ljo/dd

P.O. Box 741715
Dallas, Texas 75374-1715
Tel: 972-479-0462
Fax: 972-479-0464
November 24, 2006